# Refine Search

**Search Results -**

| Term | Documents |
|---|---|
| INDEPENDENT | 426669 |
| INDEPENDENTS | 85 |
| CONVERT$ | 0 |
| CONVERT | 244384 |
| CONVERTA | 29 |
| CONVERTABILITY | 152 |
| CONVERTABLE | 681 |
| CONVERTABLES | 2 |
| CONVERTABLY | 8 |
| CONVERTAD | 1 |
| CONVERTADAPTERACCESSKEY | 1 |
| (L23 AND (CONVERT$ WITH INDEPENDENT) ).USPT. | 10 |

There are more results than shown above. Click here to view the entire set.

**Database:**
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

**Search:** L24

Refine Search

Recall Text Clear Interrupt

## Search History

**DATE: Tuesday, March 08, 2005** Printable Copy Create Case

| Set Name side by side | Query | Hit Count | Set Name result set |
|---|---|---|---|
| | *DB=USPT; PLUR=YES; OP=ADJ* | | |
| L24 | L23 and (convert$ with independent) | 10 | L24 |

h e b b cg b e e ch

| | | | |
|---|---|---|---|
| L23 | L17 and (analy$ and append$) | 437 | L23 |
| L22 | L21 and (convert$ with independent) | 10 | L22 |
| L21 | L20 and append$ | 198 | L21 |
| L20 | L18 and analy$ | 299 | L20 |
| L19 | L18 and analy$ | 299 | L19 |
| L18 | L17 and (convert$) | 334 | L18 |
| L17 | L7 and (data with mining) | 762 | L17 |
| L16 | L14 and (document-type with independent) | 0 | L16 |
| L15 | L14 and (convert$ with independent) | 1 | L15 |
| L14 | L7 and (mining).ab. | 149 | L14 |
| L13 | L7 and (mining with dorcument$) | 0 | L13 |
| L12 | L11 and (convert$ and analy$ and (add$ or append$)) | 5 | L12 |
| L11 | L8 and (mining) | 10 | L11 |
| L10 | L8 and (convert$ with independent with format) | 2 | L10 |
| L9 | L8 and (convert$ with independent with format) | 8 | L9 |
| L8 | L7 and (extract$ with format$ with document$) | 105 | L8 |
| L7 | L6 or L1 | 29224 | L7 |
| L6 | 707/$.ccls. | 14423 | L6 |
| L5 | L3 and (extract$ with (semi-structure)) | 0 | L5 |
| L4 | L3 and (content adj2 mining) | 1 | L4 |
| L3 | 709/$.CCLS. | 17374 | L3 |
| L2 | 709/$.CCLS. | 17374 | L2 |
| L1 | 709/$.CCLS. | 17374 | L1 |

END OF SEARCH HISTORY

h e b b cg b e e ch

# Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

**Search Results** - Record(s) 1 through 10 of 10 returned.

☐ 1. Document ID: US 6750864 B1

L24: Entry 1 of 10 File: USPT Jun 15, 2004

DOCUMENT-IDENTIFIER: US 6750864 B1
TITLE: Programs and methods for the display, analysis and manipulation of multi-dimensional data implemented on a computer

Brief Summary Text (3):
This invention relates to a system, an interface to a multidimensional database, a front end to a multidimensional database, a user interface, programs and methods implemented on a digital processing unit for improving user utility of data in a multidimensional database and for integrating manipulation, mining and visualization functions of data in a multidimensional database in a unified environment.

Brief Summary Text (4):
More particularly, this invention relates to programs and methods implemented on a digital in processing unit that allow a user to display multi-dimensional data on a display device in a number of different formats to improve visualization of large amounts of data, a number of different techniques to improve data analysis and mining and a number of different techniques to improve data manipulation. These techniques provide the user with better methods for refining data selection, data categorization and/or data classification as well as providing improved visual understanding of possible relationships between variables or collections of variables in a multidimensional dataset or between multidimensional datasets. The present invention also relates to a system, an interface to a multidimensional database, a front end to a multidimensional database and a user interface to facilitate the extraction of information, interesting data relationships and meaningful data from which information can be readily derived from a multidimensional database or other similar structure containing multidimensional data.

Brief Summary Text (6):
Most multidimensional database such as MicroSoft Analysis Services, store data in a structure format. These databases perform extensive classification of data and calculations various aspects of the data as it is being stored in the database. Thus, if the database is storing information on registered voters nationwide, the data will be broken down by state, county and town or city, by gender, location, education or other classification criteria. For each such criteria, certain cumulative data associated with the raw data is calculate and stored such as the sum of each class of voter, i.e., the total number of female voter in a given town or city of a given state. The exact structure for storage of the raw data and the associated cumulative data and the mechanisms for obtaining both raw and cumulative data is understood and controlled by the database manager. The manager allows programs to access these data through specialized programing languages that correspond to database queries or requests. The form of the request is generally a

h e b b g e e e f e ef b e

set of word, symbols or functions that represent a set of instructions that the manager can invoke to obtain specific data stored in the database and transfer the requested data to the requester.

Brief Summary Text (8):
Thus, there remains a need in the art for new and better multidimensional database interfaces, front-ends, user interfaces and systems which allow for improved data visualization, analysis, mining and manipulation.

Brief Summary Text (10):
The present invention provides a computer environment for integrating multidimensional data manipulation, mining and visualization using a set of novel multidimensional data manipulation, mining and graphics techniques.

Brief Summary Text (11):
The present invention also provides an interface (sometimes referred to herein as a middleware interface or a MWI) to a MDD including a query receiver, a results sender, a query parser, a clause translator, a command sender, a data receiver and an operational construct assembler, where both sender and receiver can be combined into an exchanger and the parser and translator can be combined into a disassembler. The query receiver receives a query from a data mining technique (DMT). The parser breaks the query into an ID and one or more clauses where a clause is a syntactically valid MDD command, a pre-defined term that corresponds to a pre-defined series or sequence of syntactically valid MDD commands or a operational construct and the ID comprises a DMT identifier used by the interface and a query identifier used by the DMT. The translator translates the non command clauses into either their corresponding series of sequence of MDD commands or into a set or sequence of MDD commands necessary to satisfy the operational construct parameters. The sender sends each command to the MDD manager creating a unique synchronous thread, channel or connection to the MDD, where the MDD manager performs the necessary internal MDD procedures need to extract the requested data corresponding to each command from the MDD and once available, the MDD manager send the requested data to the receiver which receives the extracted data corresponding to each command and once received terminates that unique synchronous thread to the MDD. If the query included an operational construct, then the operational construct assembler would performs the data operations necessary to satisfy the construct parameters. The result sender would sends the results to the DMT signified in the ID associated with that query.

Brief Summary Text (12):
The present invention also provides an MDD front end including at least one data mining technique (an DMT) and an MWI of this invention. The MDD front end can also include a GUI, preferably, a GUI of this invention.

Brief Summary Text (16):
The present invention also provides data manipulation and analysis or mining techniques including at least one of the following techniques: a multidimensional decision tree generator; a cross-tab and cross-tab cell ranker (ACTG); a decision tree to cross-tab converter; a technique for identifying interesting nodes in a decision tree; a technique for constructing filters corresponding to the tree path leading to the interesting nodes; and a correlation technique.

Brief Summary Text (18):
The present invention also provides methods for visualizing, manipulating and/or analyzing or mining data implemented on a computer or digital processing unit including the step of performing at least one graphics technique of the present invention and/or at least one of the analysis technique of the present invention.

Drawing Description Text (2):
The invention can be better understood with reference to the following detailed

description together with the appended illustrative drawings in which like elements are numbered the same:

Detailed Description Text (2):
The term "MDD" means multidimensional databases such as MicroSoft Analysis Services, Hyperion Essbase, or the like.

Detailed Description Text (3):
The term "DMT" means data mining technique. .

Detailed Description Text (5):
The term "data mining technique" means one or more mathematical operations that convert raw data into information or into meaningful data.

Detailed Description Text (11):
The inventor has found that a system can be implemented on a digital processing unit to facilitate a more robust interaction with a multidimensional database (MDD), where the system includes an MDD and a front end including at least one data mining and a middleware interface (MWI) and optionally a graphics user interface (GUI). The MWI allows an dynamic asynchronous interface between query generating routines (a DMT), which are generally accessible using a GUI and a manager of an MDD. The asynchronous nature of the MWI is controlled by unique query identifiers created by each DMT when a query is sent to the MWI. The MWI then parses and translates the query into MDD commands which are forwarded the manager of the MDD via an MWI agent via establishing a unique synchronous communication channels or threads between the manager and the agent which exist until the manager send the requested data to the agent.

Detailed Description Text (13):
The present invention relates broadly to a system implements on a digital processing unit including a multidimensional database (an MDD) and an MDD front end, where the front end includes a middleware interface (an MWI) and at least one query generator or data mining technique (a DMT) and optionally a GUI of this invention.

Detailed Description Text (14):
Although several unique query generators or data mining techniques are described herein, the MWI of this invention can work in conjunction with any query generating routine. The MWI of this invention includes a query receiver adapted to receive a query from a DMT where the query includes an identifier including a DMT identifier for use by the MWI and a query identifier for use with the DMT and result sender adapted to send the results of the query back to the identified DMT. The MWI also includes a parser which breaks the query into one or more elements, where the elements are a syntactically valid MDD command, a word, phrase or symbol representing a set or series of MDD commands and

Detailed Description Text (15):
This invention allows the display, manipulation, analysis and visualization of data having multiple attributes or variables and interrelationships between such attributes and variables. To accomplish this display, manipulation and visualization, the invention uses at least one and preferably a plurality (more than one) of the following functions, routines, methods or functional aspects of this invention: (1) a scoping construct generator involving stacking 2D or 3D scatter plots to form a scope-like 3D construct to show the change in the data represented in the scatter plots relative to a sequencing variable; (2) a star construct generator involving stacking radially distributed data and connecting corresponding adjacent data pairs with surfaces; (3) a surfacing construct generator involving collection of data shown as patched surface segments in a rectangular solid; (4) a multidimensional decision tree construct generator involving displaying hierarchical value-based data as concentric rings surrounding

a central; (5) a pivot tree construct generator involving the construction of a tree representation of data where the dependent variable is a multidimensional cross-tab; (6) cross-tab pixel construct generator; (7) a multidimensional decision tree generator; (8) a cross-tab and cross-tab cell ranking routine; (9) decision tree to cross-tab conversion routine; and (10) routines for identifying interesting nodes in a decision tree and for construction the filter corresponding to the tree path leading to the interesting nodes.

Detailed Description Text (27):
Besides the general and more specific attributes and capabilities of the GUI, associated software, operating system routine and routines from other applications for display control, IO control, database access, and other similar system capabilities, the GUI's of the present invention can include any one of the following routines for the display, manipulation, analysis and visualization of multi-dimensional data.

Detailed Description Text (29):
Many of the routines that comprise part of the GUI of the present invention use a technique called a slider to aid in the display and analyze of multi-dimensional data. A slider is a type of filter or constraint which allows a user to play what-if scenarios by presetting one or more variables to a given value and analyzing the values other variables assume at that preset value of the slider variable. The slider variable gets its name because the user can adjust the preset value by sliding a slide bar in a window between the slider variable's minimum and maximum values within the data. There is no practical limit to the number of sliders variable a user can define, provided that all variables are not slider variable.

Detailed Description Text (52):
The generator also provides line fitting capabilities such as spline smoothing, least squares analysis or the like. The user can also plot statistics lines represent in the mean, median, or the like. The generator can perform shape and size statistics on the distribution of points in each scatter plot. The user can interactively select (turn on and off) one or more variable combinations or any set of variable combinations. The user can also turn on and off the connecting lines, planes outlines, cubes outlines, etc.

Detailed Description Text (80):
For example, suppose the data being analyzed relates to people and buying habits. The data is classified in a number of ways such as martial status, educational status, or the like. In addition, the data is classified in relevant time slices such as months. These classification variables are referred to as dimensions and there associated subclassifications are referred to as members. Thus, the dimension martial status has two members married and single; while the dimension educational status may have many members such as high school, college, bachelors degree, masters degree and doctorate degree, etc. Associated with each dimension and member is one or more numeric values or measures that represent different numeric data, e.g., profits, gross sales, expenses, gross revenues, etc. The star construct generator can represent data in either two or three dimensions.

Detailed Description Text (96):
Once the user selects a dimentional variable and a 2D polar plot representing the members of the dimension and their corresponding measures, the user can selected a second dimensional data and ask the technique to overlay a given statistical value so that the present polar plot can be compared to the statistical polar plot such as the mean polar plot. For example, from a given member relative to another dimension, statistics values can be calculated from the numeric value of the measure for each member, e.g., mean, median, standard deviation, etc. Once determined, a closed polygon can be constructed that represents the mean or other statistical value for members where each vertex is the mean for that particular measure of that particular member. Analogously, a closed polygon for the median

h e b b g e e e f e ef b e

points can be overlayed on each base polar plot. With visual analysis techniques such as this technique, the user can determine the following variable characteristics: (1) does the mean and/or median lie at the midpoint of the radius; (2) how does each base polar plot compare to the mean or median plot; (3) where do the statistical values appear relative to the origin; and (4) Etc.

Detailed Description Text (164):
The Automated Cross-Tab Generator (ACTG) of this invention is designed to run either on the same digital processing unit as the on-line analytical processing (OLAP) software which is normally the same digital processing unit on which the database resides or the ACTG can reside on a different digital processing unit in communication with the OLAP server.

Detailed Description Text (172):
Looking at FIG. 18A, the variables window 1666 includes a database structure window 1676 having a name column 1678, a child column 1680 and a descendant column 1682 and a must include window 1684. The name column 1678 of the structure window 1676 includes entries 1686 corresponding to levels in the database. The first entry 1688 is the database itself and has an expand/collapse button 1690 to roll up of the structure. Because these database are generally hypercubes, the next entry will be the selected cube 1692 on which ACTG will operate. The cube 1692 also has an associated expand/collapse button 1694. Each subsequent entry 1696 is a dimension of the cube 1692. The window 1662 also includes an add button 1698, a remove button 1700 and a remove all button 1702. These buttons control the addition of must variables into and out of the must variable window 1684. Each variable added to the must include window 1684 will appear with a tree state property box 1704. The property box 1704 tells ACTG whether the variable itself, its children or its descendants are to be included in the analysis. The window 1662 also includes a reset button 1706, a retain settings button 1708, a run button 1710, a cancel button 1712 and an estimated cross-tab indicator field 1714. The user can select as a variable a dimension, a member of a dimension, the children or a dimension or a member or all the descendants of dimension or a member. If the user selects a dimension or a member, then a "v" labels is associated with the selection to indicate that the selected dimension or member is a variable. If the user selects the children of a dimension or member, then the selected dimension or member is shown with a "c" label. If the user selects all descendants (the full hierarchy), then the selected dimension or member is shown with an "h" label.

Detailed Description Text (173):
Looking at FIG. 18B, the limits window 1670 includes a set of limit definition toggles 1716 and a nulls (%) level selection box 1718. The null level spin selection box 1718 tells ACTG how to handle sparely populated cross-tabs. The process involves calculating the percentage of null cells per row, column, and overall. The user can specify the percentage of sparsity from 0% to 100%. A 0% sparsity would mean that the cross-tab has sparsity and only cross-tabs with no sparsity would be analyzed; while a 100% sparsity would mean all cross-tabs are analyzed regardless of its sparsity. The row percent sparsity is calculated by dividing the total number of sparse cells in a row by total of cells int the row. The column percent sparsity is calculated by dividing the total sparse cells in a column by the total number of cells in the column. The overall percent sparsity will be calculated by dividing the total sparse cells in a cross-tab by total number of cells in a cross-tab.

Detailed Description Text (174):
When the user set a given null % level in the selection box 1718, all cross-tabs with a any one of the sparsity measures greater than the limit are not analyzed. That is, if a cross-tab an overall sparsity greater than the cutoff, it is not analyzed; if the cross-tab has a row sparsity for any row that is greater than cutoff, it is not analyzed, and finally if the cross-tab has a columns sparsity for any column greater than the cutoff, it is not analyzed. In other words, for a

cross-tab to be analyzed every sparsity measure, overall, row-by-row and column-by-column must be less than or equal to the cutoff.

Detailed Description Text (175):
Looking at FIG. 18C, the advanced window 1672 a set of advanced control buttons 1720 allow the user to further control the analysis.

Detailed Description Text (176):
The setting controls allow a user to spin control limit values, edit control the number of high cross-tabs to be shown, check control low limits (a user may not be interested in the low cross-tabs), spin edit control a low limit, edit box control the number of low cross-tabs to be shown; and to designate the site for running the analysis: run local or remote.

Detailed Description Text (177):
As stated above, the user can define/specify the variables to be included in or excluded from an analysis. To make any variable a part of the analysis, the user can simply click to check the box associated with that variable and its descendants will be included in the analysis. Conversely, the user can click inside a checked box to uncheck it. Any unchecked variable will not be considered to be a part of the cross-tab. The count of the variables in a dimension is also shown so that the user has an idea of how many variable have been selected.

Detailed Description Text (178):
The user can also force ACTG to include one or more variables in each cross-tab, i.e., each cross-tab has to contain the "must include" variable. As the user selects variables to be included in the analysis, the estimated number of cross-tab is calculated and displayed.

Detailed Description Text (182):
Referring now to FIG. 19, an application window 1910 of this invention is shown to includes a tool bar 1912, a database structure window 1914 equivalent to the variables window associated with the variables tab of the settings window, a ACTG results window 1916, a cross-tab display window 1918 and a cross-tab graphics visualization window 1920. Once the results of an ACTG analysis are complete, the user can look at the interesting cross-tabs and their corresponding values. The user selects an entry of this list of interesting cross-tabs and drags and drops the selected entry into the cross-tab window 1918. The cross-tab is then constructed or retrieved from its entry description. In the figure, the top most entry has been dragged and dropped into the window. The user can then select to view the cross-tab data visually using any visualization technique in the main application program. In the figure, the user choose a histogram.

Detailed Description Text (186):
Likewise the user can exclude one or more variables to be considered by ACTG. For example, the user can say: do not include country=Canada or country=Mexico as a variable when analyzing the data.

Detailed Description Text (203):
The process of generating a cross-tab and calculating the deviation is a self-contained method, i.e. it does not require any user intervention. This process can run on a server, create a ranked (by deviation) list of cross-tabs. A user can login to the server, and view the cross-tab list that have been generated by the server so far. If he is interested in a particular cross-tab listing, then he can drag-n-drop it upon the cross-tab control in the analysis pane. Once the cross-tab control is populated the user is free to do whatever he pleases.

Detailed Description Text (218):
The ACTG DLL is where the main data mining algorithms and data management coding resides. This DLL can request data directly from the OLAP Database and performs its

analysis or preferably, this DLL request data indirectly through the middleware interface of this invention. The OLAP database may be local or remote. Also note, that the ACTG DLL is an ordinary Win32 DLL rather than a COM server. Functions exported from the DLL are used to connect to the database, and calculate the cross tabs.

Detailed Description Text (220):
For higher performance and scalability, it may be desirable to place the ACTG DLL code on a remote machine with the OLAP Database. In this case, it is desirable that the calculations and data mining activities occur on another machine, and that results may be stored for access by multiple application clients. In this scenario, the database server is host to the ACTG Service. The ACTG Service loads the ACTG DLL directly and makes use of it for data mining and determining a list of interesting cross tabs. This service (which is started when NT starts on the host machine) offers several transparent features over the configuration previously described configuration.

Detailed Description Text (221):
The first of which is that the ACTG calculations are saved (perhaps to a scratch file) and can be shared among multiple application clients on the network. This avoids having to recalculate the same ACTG tables each time the algorithm is used. The ACTG Service is informed that the database has changed (and thus it is time to refresh its tables) via a Stored Procedure on the database server that is triggered whenever a new database dump is performed. The ACTG Service can run at idle priority and can perform its data mining activities when the server is not busy. If a client connection is made before a complete list of cross tabs is generated, then the priority of the service can be temporarily boosted to allow completion of its data mining.

Detailed Description Text (228):
The breath-first process involves building the cross-tab by generating the dimension/member combinations at the highest level of each variable, followed by the next level and so on. This process is generally preferred because the more interesting cross-tab usually occur higher up in the variable hierarchies, but becomes particularly preferred when the sparsity cutoff is less than 100%. When the sparsity cutoff is set a some value say 20%, then as the process goes do the level of variable hierarchy analyzing the cross-tabs an each hierarchical level, when a cross-tab fails the sparsity test, then the entire branch can be ignored. Thus, significantly shortening computational time and resource costs.

Detailed Description Text (240):
Another powerful feature of the cross-tab node based decision tree processing of this invention comes from the different types of variables represented in a data structures amenable to decision tree processing. In particular, the data structures being analyzed are generally hierarchical in nature. Thus, a given dependent variable in any given cross-tab node may be a hierarchical variable. Because a dependent variables may be hierarchical, the user can use standard hierarchical drill up and drill down functions on each node that includes a hierarchical variable. This function is invoked simply by selecting a node and then click or double clicking on the hierarchical variable. Preferably, hierarchical variables are indicated by color or marked in some other fashion or all hierarchical variables will have active drill up or drill down button associated therewith.

Detailed Description Text (264):
Pivot tree (an enhanced decision tree) is a typical example of a machine learning algorithm that can effectively work within an OLAP environment. A classical decision tree generator builds a decision tree by individually analyzing all detail data records in a database. Entire database scan (read) operations are performed very frequently. Whereas, the pivot tree generator of this invention, takes advantage of statistical calculations that an OLAP engine performs at data

load/refresh time. The pivot tree generator can extract all necessary information, to grow a tree, from an nD cross-tab (that can be queried from the OLAP database manager by using middleware). In pivot tree generation, detail data records are not processed at all. Thus, the processing time to create a pivot tree is drastically reduced.

Detailed Description Text (265):
In a similar way, other data mining algorithms such as feature selection, association rules, clustering etc. can utilize the OLAP environment and Middleware interface to speed up the processing.

Detailed Description Text (280):
DATA MINING, MIDDLEWARE AND OLAP ENVIRONMENT

Detailed Description Text (282):
In order to efficiently use data-intensive operations that are common to classifiers, decision trees and other machine learning and pattern recognition algorithms or data mining techniques used in Data mining of data in a multidimensional database (MDD), the present invention introduces an interface between such application programs and an MDD called a Middleware interface (MWI) that allows efficient command sending and data receiving from an MDD using a Database Manager (such as MicroSoft on line analytical processing "OLAP" engine) of a MDD. For additional information on MicroSoft's OLAP databases, the reader is directed to www.microsoft.com and the section on OLAP databases.

Detailed Description Text (283):
In some data mining algorithms or techniques, direct access to the data is not absolutely necessary, and only some statistics about the data is required. Even in the cases where direct access to the data is needed, some information can be efficiently obtained using the Database manager directly. However, to improve flow of command and data to and from the MDD manager, the MWI of this invention is ideally suited.

Detailed Description Text (284):
The purpose of the MWI of this invention is to serve as an interface between the data mining algorithms or techniques (DMTs) and the MDD manager. The MWI includes all the functions needed to receiver queries from a plurality of DMTs, determine the MDD commands required to satisfy the request, schedule and batch submit the commands to the MDD manager and receive data from the MDD manager and match the data with the appropriate command and then forward the data to the requesting DMT in the form mandated by the query. The MWI, thus, is a staging platform between the DMTs and the MDD Manager to improve data retrieval from the MDD Manager. The MWI is asynchronous when interacting with DMTs and uses multiple threads when interacting with the MDD manager, allowing parallel processing, staging, to occur improving overall throughput of the volume of command/data exchanges.

Detailed Description Text (285):
The MWI of this invention interacts directly with the OLAP Data warehouses feature of the MDD Manager. The present invention includes a set of classification algorithms or DMTs associated with an NPI data mining library such as a C++ library which a user can invoke as data analysis tool for extracting information or meaningful data from a MDD. The Middleware interface of this invention acts as a layer between the DMTs and the MDD which allows for more abstract or complex data retrieval and storage services derived from queries from a data mining algorithm or DMT.

Detailed Description Text (300):
Data Mining Middleware: OLAP Datasource and DMTs

Detailed Description Text (302):

h e b b g e e e f e ef b e

One aspect of the system of the present invention is the construction of an interface between data mining algorithms or DMTs and an multidimensional database (MDD) having a sophisticated OLAP type engine. The such an environment, DMTs have a greater flexibility in function if the can offload much of the MDD interaction to middle man that can take full advantage of the considerable functionality built in to an OLAP engine or other similar engines associated with MDDs such as their Data warehousing capabilities. The system of the present invention accomplishes this offloading via the middleware interface (MWD. Using the MWI, the system of this invention can run the DMTs as member of an NPI data mining C++ library as data analysis tools from a menu or set of icons on a GUI.

Detailed Description Text (303):
The MWI of this invention has at least the following features: (1) the MWI allows DMTs to be designed independent of the logical and physical data structure of any particular MDD; (2) the MWI performs all needed data conversions; (3) the MWI performs needed data pre-processing depending on the data, meaningful data or information requested by a particular DMT; (4) the MWI performs all data request scheduling; (5) the MWI performs all optimization of data request scheduling, converting and pre-processing; (6) the MWI establishes necessary multiple threads (multithreading) to satisfy the data, meaningful data or information requested by a particular DMT; and (7) the MWI is an asynchronous interface capable of simultaneously processing multiple DMTs queries.

Detailed Description Text (313):
This document presents a technical discussion of the Middleware major features. The Middleware provides Data Mining developers with an advanced data access application with analytical processing capabilities implemented as a COM-server.

Detailed Description Text (318):
The MWI of this invention includes a OLAP component and of course also includes a RDB component. These components are interfaces between data sources and data mining algorithms. The RDB MWI executes an SQL-query by means of SQL RDBMS. Given the resulting record set, RDB MWI component calculates some statistics (frequency tables) that are useful for efficient data mining. With certain OLAP services, the OLAP MMI component does not process the data by itself. OLAP MWI component is a superstructure over the multidimensional data source and all the essential features of data processing are available for DMTs by means of the MWI. One important difference between the MWI of this invention and the standard DMT--MDD interaction process is that the data is presented in terms of variables (class or dependent variables and independent ones) and it's values. Thus, the MWI is useful in data mining and machine learning in an MDD environment.

Detailed Description Text (326):
The MWI allows each client DMT to retrieve some part of needed data in an analyze form, where the analysis is performed at the MWI level prior to being forwarded to the DMT for final compilation and processing and ultimately to a visualizer for presentation and manipulation by the user. The MWI handles all necessary MDD commands and queries, while the DMT is free to perform other tasks. The MWI allows the DMT to construct queries at a higher level of abstraction than the MDD structural level generally supports, where the queries are defined in terms of variables (data mining native format). The MWI then posts the results when all data retrieval operations are completed. The dataset access component of the MWI is responsible for any data exchange with the DMT.

Detailed Description Text (338):
All DMTs work in MicroSoft's Component Object Module (COM) framework. For additional information on MicroSoft's Component Object Module technology, the reader is directed to www.microsoft.com and the section on COM-based technologies. In other words, the DMT have two components dealing with the one data mining method. The first component is a NPI C++ object and the second component is a COM

h e b b g e e e f e ef b e

object. Referring now to FIG. 34, a pictorial representation of a DMT component structure for use with the MWI of this invention 3410 is shown to include a C++ or code object 3412 and a COM object 3414. The pictorial 3410 also shows an external connection to another object 3416.

Detailed Description Text (341):
All COM objects related to Data mining are expected to be located in an OlapDm.DLL, but NPI C++ classes may be used elsewhere.

Detailed Description Text (346):
The correlation DMT is a DMT that takes specific advantage of the MicroSoft Analysis Services Correlation function invokable at the OLAP engine level. Using the Correlation DMT, a correlation scenario can defined by the user, sent to the MWI, process by the MWI as commands to the MDD to perform one or more MicroSoft correlation functions. For example, suppose the user has constructed the cross-tab shown in FIG. 36. Looking at FIG. 36, a cross-tab 3610 is shown having three dimension 3612, 3614 and 3616, USA, Drink and Food, respectively. The USA dimension 3612 includes one state member, OR, 3618 and the state member include a number of city member 3620. The Drink dimension 3614 includes a number of drink members 3622 and the Food dimension includes a number of food 3624. The cross-tab 3610 is populated with corresponding measure values 3626.

Current US Cross Reference Classification (1):
707/4

CLAIMS:

5. A method implemented on a digital process unit for analyzing data in a multi-dimensional dataset comprising the steps of: a. selecting n variables from a multidimensional dataset, where n is an integer less than or equal to the dimensionality of the dataset; b. selecting a cross-tab dimension, m, where m is an integer having a value less than or equal to n or having a range of values between a lower limit greater than to equal to 1 and an upper limit less than or equal to n; c. constructing k cross-tabs of dimension m, where k is the number of combinational cross-tabs derived from n variable taken m at a time; and d. do ranking; e. displaying a list of the ranked cells with cross-tab identification information; f. selecting a desired cell from the list; g. display the corresponding cross-tab with highlight cell.



☐ 2. Document ID: US 6711585 B1

L24: Entry 2 of 10 File: USPT Mar 23, 2004

DOCUMENT-IDENTIFIER: US 6711585 B1
TITLE: System and method for implementing a knowledge management system

Abstract Text (1):
A method and system organize and retrieve information using taxonomies, a document classifier, and an autocontextualizer. Documents (or other knowledge containers) in an organization and retrieval subsystem may be manually or automatically classified into taxonomies. Documents are transformed from clear text into a structured

h e b b g e e e f e ef b e

record. Automatically constructed indexes help identify when the structured record is an appropriate response to a query. An automatic term extractor creates a list of terms indicative of the documents' subject matter. A subject matter expert identifies the terms relevant to the taxonomies. A term analysis system assigns the relevant terns to one or more taxonomies, and a suitable algorithm is then used to determine the relatedness between each list of terms and its associated taxonomy. The system then clusters documents for each taxonomy in accordance with the weights ascribed to the terms in the taxonomy's list and a directed acyclic graph (DAG) structure is created.

Brief Summary Text (10):
Documents stored in the organization and retrieval subsystem may be manually through an attribute matching process or automatically classified into a predetermined number of taxonomies through a process called autocontextualization. In operation, the documents are first transformed from clear text into a structured record (knowledge container) automatically constructed indexes (tags) to help identify when the structured record is an appropriate response to a particular query. An automatic term extractor creates a list of terms that are indicative of the subject matter contained in the documents, and then a subject matter expert identifies the terms that are relevant to the taxonomies. A term analysis system assigns the relevant terms to one or more taxonomies, and a suitable algorithm is then used to determine the relatedness (weight) between each list of terms and its associated taxonomy. The system then clusters documents for each taxonomy in accordance with the weights ascribed to the terms in the taxonomy's list and a directed acyclic graph (DAG) structure is created.

Brief Summary Text (12):
Additional features and advantages of the invention will be set forth in the description which follows, and in part will be apparent from the description, or may be learned by practice of the invention. The objectives and other advantages of the invention will be realized and attained by the methods, systems, and apparatus particularly pointed out in the written description and claims hereof, as well as the appended drawings.

Detailed Description Text (86):
The input into the knowledge map generation mechanism is a set of documents and a set of "target" taxonomy root nodes. The output is a knowledge map. A set of steps and algorithms that translate the former into the latter is described below. The starting point for knowledge map generation, as shown in FIG. 9, is the collection of documents that will be managed by the e-Service Portal (step 902). This collection will be referred to as the generation corpus. The generation corpus must either be the basis for the knowledge containers to be used within the Portal or is representative of the content to be placed in knowledge containers. In one embodiment, the generation corpus has the following characteristics: (1) the documents in the corpus are a statistically valid sample of the documents to be managed; (2) there are at least 1,000 and less than 30,000 documents; (3) there are at least the equivalent of 500 pages of text and no more than 50,000 pages of text; and (4) the documents are decomposable into ASCII text. The knowledge map generation process described below is language independent. That is, so long as the documents can be converted into electronic text, the process is also independent of document format and type.

Detailed Description Text (89):
As stated earlier, the described process generates a knowledge map. There is one taxonomy for each root concept-node in the input set. As shown in FIG. 9, the first step (904) is document collection. The generation corpus is a representative sample of documents from a single coherent knowledge domain, the representation of which meets the needs of a specific business problem or domain. In one typical scenario, an enterprise has a corpus of documents over which they would like to provide the retrieval and display capabilities described earlier in this specification. In that

case, the generation corpus would be a subset of the enterprise's corpus of documents. The subset may be manually identified. In another scenario, the knowledge domain is well-defined, but the enterprise does not yet have a corpus covering the domain. In this case, representative documents must be found and accumulated to form the generation corpus If the available corpus is larger than the maximum size prescribed above, sampling procedures may be employed to choose a subset of documents for use in the generation corpus. As shown in step 906, the next step is to convert the documents into XML marked text as described above in the portion of the document that addressed autocontextualization. Next, in step 908, the system performs root concept-node collection and input. A set of root concept nodes is provided, with the following information about each: taxonomy name (examples are "Geography", "Industry", and "Business Topic"); root node name (examples are "The World", "Private Sector" and "The Business World"); root identifier (any string unique within the set); and domain name (a unique string common to all root concept-nodes within the knowledge map). In a preferred embodiment, a file is prepared designating the set of root concept-nodes. This file is provided as an input to knowledge map generation and includes one record (with all associated information) for each root. Next, in step 910, the system identifies and inputs the generation corpus. In one embodiment, a file listing each individual document in the generation corpus and its physical location, one per line, is provided as an input to knowledge map generation. In step 912, term extraction is then performed. Using any valid algorithm for term feature extraction, a list of corpus terms is generated. The term list is ordered by frequency or weight. This term list includes all words and multiple word combinations deemed to have statistical significance as indicators of meaning within the generation corpus. The term list is a function of the generation corpus documents--the text of these documents is read and parsed to produce the list. A term may have any (or none) of the following characteristics in any combination: a term may be case-sensitive (the term "jaguar" is distinct from the term "Jaguar"; a term may be one or more words ("lion" or "Barbary lion" or "South Barbary lion"); a term may include punctuation ("INC." or "Yahoo!"); or a term may be a form of markup ("<NAME> John Smith</NAME>"). In step 914, the system then performs term separation. Terms are presented to a subject matter expert (SME) highly familiar with the knowledge domain associated with the generation corpus. The SME designates whether the term is relevant to each of the taxonomies in the input set. Each term may be relevant in zero to N taxonomies where N is the number of root concept-nodes. For example, the term "jaguar" may be relevant to the taxonomy on "Mammals" and the taxonomy on "Automobiles". The result of this step is N lists of terms where N is equal to the number of root concept-nodes. In one embodiment, the SME generates a set of terms a priori, from his or her knowledge of the domain, for each root concept node. The terms extracted in step 912 are automatically provisionally designated as relevant to zero or more taxonomies according to their similarity to the SME-generated term sets, using any word-similarity measures or algorithms from the fields of computational linguistics and information retrieval. These designations are presented to the SME for validation. Next, in step 916, the system performs term analysis. In that step, a report is generated with the following information: (1) the number (raw count) of terms assigned to each taxonomy; (2) the Pearson correlation between the terms assigned to each taxonomy with the terms assigned to every other taxonomy; and (3) a list of terms assigned to each taxonomy ordered by weight or frequency. Processing then flows to step 920, where the system performs diagnosis for irrelevant root concept nodes. In step 922, the system determines whether any taxonomy is assigned a small number or percentage of the term/features. If there are taxonomies that are assigned to a small number of terms/features, processing flows to step 924 and the concept node is removed from the input list. Processing then flows to step 908 and the process repeated. The system in step 926 then conducts a diagnosis for overlap and diagnosis for non-orthogonality. If the terms ascribed to any taxonomy correlate to a very high degree with the terms ascribed to any other taxonomy, then the taxonomies in question may overlap (step 926). In the case of overlap, one or more of the root concept-nodes with a high cross-correlation should be eliminated (step 928). Processing then flows to step

908 and the entire process repeated. Such high correlation of terms may alternatively indicate that the taxonomies in question are non-orthogonal (step 930). In this case, a set of two or more of the root concept-nodes with a high cross-correlation should be replaced with a more abstract root concept-nodes (step 932). Processing then flows to step 908 and the process repeated. If the system determines that there is not overlap or non-orthogonality, processing flows to step 934, where term weighting is performed. Using any standard algorithm for weighting a list of features in terms of relative importance, the term list for each taxonomy is weighted. Terms have a unique weight in relationship to each taxonomy to which they are ascribed. So, the term "jaguar" may have a low weight in relationship to the "Mammal" taxonomy and a high weight in relationship to the "Automobile" taxonomy and a zero weight (non-ascribed) in relationship to a "Geography" taxonomy. Optionally, the system may in step 936, subject the term weights generated in step 934 to review by an SME. The SME may then enter a new weight, replacing the computer-generated weight. One weighting algorithm has the following key characteristics:

Current US Original Classification (1):
707/104.1

Other Reference Publication (22):
IBM's Data Mining Technology (1996).



---

☐ 3. Document ID: US 6640249 B1

L24: Entry 3 of 10 File: USPT Oct 28, 2003

DOCUMENT-IDENTIFIER: US 6640249 B1
**** See image for Certificate of Correction ****
TITLE: Presentation services patterns in a netcentric environment

Detailed Description Text (2):
A preferred embodiment of a system in accordance with the present invention is preferably practiced in the context of a personal computer such as an IBM compatible personal computer, Apple Macintosh computer or UNIX based workstation. A representative hardware environment is depicted in FIG. 1, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit 110, such as a microprocessor, and a number of other units interconnected via a system bus 112. The workstation shown in FIG. 1 includes a Random Access Memory (RAM) 114, Read Only Memory (ROM) 116, an I/O adapter 118 for connecting peripheral devices such as disk storage units 120 to the bus 112, a user interface adapter 122 for connecting a keyboard 124, a mouse 126, a speaker 128, a microphone 132, and/or other user interface devices such as a touch screen (not shown) to the bus 112, communication adapter 134 for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter 136 for connecting the bus 112 to a display device 138. The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present invention may also be implemented on platforms and operating systems other than those mentioned. A preferred embodiment is written using JAVA, C, and the C++language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to

h e b b g e e e f e ef b e

develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions require adaptation to make use of the benefits of OOP. A need exists for these principles of OOP to be applied to a messaging interface of an electronic messaging system such that a set of OOP classes and objects for the messaging interface can be provided. OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.

Detailed Description Text (34):
Architecture also is an engineering discipline. It creates and also depends on a structured manner to analyze and design whatever is to be built. Like all living disciplines, architecture continues to grow and evolve. Engineering discoveries move the field forward. Certain design and engineering principles clearly show themselves to be successful in practice, and these then become repeatable components of additional work. The ability to continue to master each component, as well as the interrelations among components, is a distinguishing characteristic of architecture.

Detailed Description Text (39):
Finally, architecture must be understood as a process 200, not just a thing. This process can be described at a very high level using FIG. 2. Step 1: Analyze 202. The architect must begin by listening to and researching the needs of the client. What is the function of the building? What is its environment? What are the limitations set by budget and use? Step 2: Design 204. This is a blueprint stage. The architect creates one or several designs showing the layout of the structure, how different spaces fit together, how everything looks from different views, what materials are to be used, and so forth. Step 3: Model & Test 206. Not every architectural project has this step, but in many cases, the architect will create a scale model/prototype of the finished product, allowing the client a clearer sense of what the ultimate solution will look like. A model is a kind of test stage, allowing everyone to test the design in a near-real-life setting. Step 4: Build 208. This is the actual construction of the building, in general accord with the blueprints and prototype. Step 5: Operate and Evolve 210. The building is to be lived in and used, of course, and so an important step is to ensure that the finished product is tended and operated effectively. Architects themselves may not be involved in the operation of their building, but they certainly would be involved in future expansions or evolutions of the building. Stewart Brand's recent text, How Buildings Learn, argues that effective architecture takes into account the fact that buildings "learn": as people live and work in them over time, those people will seek to alter the building in subtle, or not so subtle, ways.

Detailed Description Text (73):
The use of architecture frameworks during analysis and design can reduce the risks of an IT solution. It should improve development productivity through reuse, as well as the IT solution's reliability and maintainability.

Detailed Description Text (135):
Existing Architecture and Infrastructure 700 E1. Other Netcentric applications been developed and placed in production. The user community is often less resistant to accept the use of new technology to address changing business drivers if they are not completely unfamiliar with the characteristics of the technology. If an application based on a Netcentric architecture has already been successfully

h e b b g e e e f e ef b e

piloted or deployed, acceptance of additional systems will be eased. E2. The client has significant technology skills within its IT department. This is especially important if the client plans on developing or operating the application themselves. A significant investment in training and changes to internal organizations may be necessary for successful deployment of this type of system. The client must have a culture that supports change. Some organizations are very conservative and strong, making it difficult to deliver a successful project using new technology. E3. The client has multiple hardware/operating system configurations for their client machines. In traditional client/server environments, distributing an application internally or externally for an enterprise requires that the application be ported, recompiled and tested for all specific workstation operating systems. Use of a Universal Client or web-browser may eliminate many of these problems by providing a consistent and familiar user interface on many different operating systems and hardware platforms. E4. The application will run on a device other than a PC. The momentum of the Internet is putting a lot of pressure on vendors of various devices to be web-enabled. Having the Internet infrastructure in place makes it more feasible for vendors to create new physical devices from which electronic information can be accessed. For example, Web televisions are gaining momentum. Now users can access the Internet from a television set. Network Computers, thin-client devices that download and run applications from a centrally maintained server are generating a lot of interest. Also, users want to have access to the same information from multiple physical devices. For example, a user might want to have access to his/her e-mail from a cellular phone, from a Web TV or their portable PC. E5. The current legacy systems can scale to serve a potentially large new audience. Expanding the user community of a legacy host or client/server system by including an audience which is external to the company can result in dramatic increases in system usage. The additional demand and increased usage placed on existing legacy systems is often difficult to estimate or predict. Analysis must be conducted to ensure existing legacy systems and infrastructure can absorb this increase.

Detailed Description Text (519):
In summary, the Directory service performs the following functions: Stores information about network resources and users and tracks relationships Organizes resource access information in order to aid resources in locating and accessing other resources throughout the network Provides location transparency, since resources are accessed through a directory rather than based on their physical location Converts between logical resource names and physical resource addresses Interacts with Security services such as authentication and authorization track identities and permissions Provides single network logon to file and print resources; can provide single network logon for network applications that are integrated with the Directory service Distributes directory information throughout the enterprise (for reliability and location-independent access) Synchronizes multiple directory databases Enables access to heterogeneous systems (integration of various network operating systems, platforms, etc.)

Detailed Description Text (591):
The following are examples of File Transfer products: Computer Associates' CA-XCOM--provides data transport between mainframes, midrange, UNIX, and PC systems. XcelleNet's RemoteWare--retrieves, appends, copies, sends, deletes, and renames files between remote users and enterprise systems. Hewlett-Packard's HP FTAM--provides file transfer, access, and management of files in OSI networks.

Detailed Description Text (746):
Authentication can occur through various means: Basic Authentication--requires that the Web client supply a user name and password before servicing a request. Basic Authentication does not encrypt the password in any way, and thus the password travels in the clear over the network where it could be detected with a network sniffer program or device. Basic authentication is not secure enough for banking applications or anywhere where there may be a financial incentive for someone to

steal someone's account information. Basic authentication is however the easiest mechanism to setup and administer and requires no special software at the Web client. ID/Password Encryption--offers a somewhat higher level of security by requiring that the user name and password be encrypted during transit. The user name and password are transmitted as a scrambled message as part of each request because there is no persistent connection open between the Web client and the Web server. Digital Certificates or Signatures--encrypted digital keys that are issued by a third party "trusted" organization (i.e. Verisign); used to verify user's authenticity. Hardware tokens--small physical devices that may generate a one-time password or that may be inserted into a card reader for authentication purposes. Virtual tokens--typically a file on a floppy or hard drive used for authentication (e.g. Lotus Notes ID file). Biometric identification--the analysis of biological characteristics to verify individuals identify (e.g., fingerprints, voice recognition, retinal scans).

Detailed Description Text (806):
QoS can be achieved in various ways as listed below: Specialized QoS Communications Protocols--provide guaranteed QoS. Asynchronous Transfer Mode (ATM)--ATM is a connection-oriented wide area and local area networking protocol that delivers QoS on a per-connection basis. QoS is negotiated as part of the initial connection set up and as network conditions change. Because of the small size of ATM data cells, QoS can be better managed, compared to protocols such as Ethernet that have large frames that can tie up network components. For ATM to deliver QOS to applications, ATM must be used end-to-end. Resource Reservation Protocol (RSVP)--The emerging RSVP specification, proposed by the Internet Engineering Task Force (IETF), allows applications to reserve router bandwidth for delay-sensitive IP traffic. With RSVP, QoS is negotiated for each application connection. RSVP enables the network to reserve resources from end to end, using Frame Relay techniques on Frame Relay networks, ATM techniques on ATM, and so on. In this way, RSVP can achieve QoS across a variety of network technologies, as long as all intermediate nodes are RSVP-capable. IP Stream Switching--improves network performance but does not guarantee QoS. IP Switching--IP Switching is an emerging technology that can increase network throughput for streams of data by combining IP routing software with ATM switching hardware. With IP Switching, an IP switch analyzes each stream of packets directed from a single source to a specific destination, and classifies it as short- or long-lived. Long-lived flows are assigned ATM Virtual Channels (VCs) that bypass the IP router and move through the switching fabric at the full ATM line speed. Short-lived flows continue to be routed through traditional store-and-forward transfer. Tag Switching--Like IP Switching, emerging Tag Switching technology also improves network throughput for IP data streams. Tag Switching aggregates one or more data streams destined for the same location and assigns a single tag to all associated packets. This allows routers to more efficiently transfer the tagged data. Tag Switching is also known as Multiprotocol Label Switching. Data Prioritization--improves network performance but does not guarantee QoS. While not an example of end-to-end QoS, various network components can be configured to prioritize their handling of specified types of traffic. For example, routers can be configured to handle legacy mainframe traffic (SNA) in front of other traffic (e.g., TCP/IP). A similar technique is the use of prioritized circuits within Frame Relay, in which the Frame Relay network vendor assigns different priorities to different permanent virtual circuits. Prioritization techniques are of limited effectiveness if data must also pass through network components that are not configured for prioritization (e.g., network components run by third party network providers).

Detailed Description Text (945):
Logging must be done, however to mitigate problems, centralize logs and create a standard, usable log format. 3rd party logs should be mapped into the central format before any analysis is attempted.

Detailed Description Text (1076):

h e b b g e e e f e ef b e

Workflow can be further divided into the following components: Role management Role management ie provides for the assignment of tasks to roles which can then be mapped to individuals. A role defines responsibilities which are required in completing a business process. A business worker must be able to route documents and folders to a role, independent of the specific person, or process filling that role. For example, a request is routed to a supervisor role or to Purchasing, rather than to "Mary" or "Tom." If objects are routed to Mary and Mary leaves the company or is reassigned, a new recipient under a new condition would have to be added to an old event. Roles are also important when a number of different people have the authority to do the same work, such as claims adjusters; just assign the request to the next available person. In addition, a process or agent can assume a role; it doesn't need to be a person. Role Management Services provide this additional level of directory indirection. Route management Route management enables the routing of tasks to the next role, which can be done in the following ways: Serial--the tasks are sequentially performed; Parallel--the work is divided among different players; Conditional--routing is based upon certain conditions; and Ad hoc--work which is not part of a predefined process. Workflow routing services route "work" to the appropriate workflow queues. When an application completes processing a task, it uses these services to route the work-in-progress to the next required task or tasks and, in some cases, notify interested parties of the resulting work queue changes. The automatic movement of information and control from one workflow step to another requires work profiles that describe the task relationships for completing various business processes. The concept of Integrated Performance Support can be exhibited by providing user access to these work profiles. Such access can be solely informational--to allow the user to understand the relationship between tasks, or identify which tasks need to be completed for a particular work flow--or navigational--to allow the user to move between tasks. Route Management Services also support the routing and delivery of necessary information (e.g., documents, data, forms, applications, etc.) to the next step in the work flow as needed. Rule Management A business process workflow is typically composed of many different roles and routes. Decisions must be made as to what to route to which role, and when. Rule Management Services support the routing of workflow activities by providing the intelligence necessary to determine which routes are appropriate given the state of a given process and knowledge of the organization's workflow processing rules. Rule Management Services are typically implemented through easily maintainable tables or rule bases which define the possible flows for a business event. Queue Management These services provide access to the workflow queues which are used to schedule work. In order to perform workload analysis or to create "to do lists" for users, an application may query these queues based on various criteria (a business event, status, assigned user, etc.). In addition, manipulation services are provided to allow queue entries to be modified. Workflow services allow users and management to monitor and access workflow queue information and to invoke applications directly.

Detailed Description Text (1100):
Issues to consider include the following: (1) samples and assists that are available to the developer; (2) existence of a scripting or programming language; (3) granularity of the security, or in other words, at what levels can security be added; (4) freedom of choosing productivity applications; (5) existence of aggregate functions which allow for analysis of the workflow efficiency; (6) existence/need for Business Processing Re-engineering tools.

Detailed Description Text (1143):
Objects are an easy metaphor to understand and manage. There are still substantial risks involved, particularly because component- and object-orientation has a pervasive impact on areas as broad as analysis and design, planning, and development tools.

Detailed Description Text (1146):
Component and object technology impacts most aspects of software development and

h e b b g e e e f e ef b e

management. Component technology is a new technology and a driving influence in the evolution of object-oriented (OO) methodologies. The Management Considerations section of the Introduction to Component-Based Development uses the Business Integration (BI) Model to discuss the impact of OO, including: Strategy and planning with a long-term view towards building reusable, enterprise software assets. Technology and architecture approaches for building cohesive, loosely coupled systems that provide long-term flexibility. Processes that shift analysis/design techniques from functional, procedural decomposition to business process modeling. These techniques are then used to decompose the system into domain objects and processes. People and organization strategies that emphasize greater specialization of skills within structures that support inter-team collaboration.

Detailed Description Text (1149):
Many of these considerations have been addressed over the last few years. Most published literature continues to focus on narrow technology issues, such as programming techniques or generic methodologies, such as analysis and design approaches or notation. Still, a growing number of publications and vendor strategies attack the enterprise needs within on-line netcentric execution models. Real-world, client solutions involve making pragmatic decisions, in which compromise occurs at the intersection of the four major OO themes. Experience with many component client projects in diverse industries uniquely positions a user to effectively address these complexities.

Detailed Description Text (1167):
Business Components represent real-world concepts in the business domain. They encapsulate everything about those concepts including name, purpose, knowledge, behavior, and all other intelligence. Examples include: Customer, Product, Order, Inventory, Pricing, Credit Check, Billing, and Fraud Analysis. One might think of a Business Component as a depiction or portrait of a particular business concept, and as a whole, the Business Component Model is a depiction or portrait of the entire business. It's also important to note that although this begins the process of defining the application architecture for a set of desired business capabilities, the applicability of the Business Component Model extends beyond application building.

Detailed Description Text (1172):
In the Business Architecture stage 3604, a project team begins to define the application architecture for an organization's business capabilities using Business Components. Business Components model real-world concepts in the business domain (e.g., customers, products, orders, inventory, pricing, credit check, billing, and fraud analysis). This is not the same as data modeling because Business Components encapsulate both information and behavior. At this point in the process, an inventory of Business Components is sufficient, along with a definition, list of entities, and list of responsibilities for each Business Component.

Detailed Description Text (1173):
In Capability Analysis 3606 and the first part of Capability Release Design 3608, the project team designs Business Components in more detail, making sure they satisfy the application requirements. The team builds upon its previous work by providing a formal definition for each Business Component, including the services being offered. Another name for these services is "Business Component Interfaces." The team also models the interactions between Business Components.

Detailed Description Text (1181):
Business Components on the process-centric side of the spectrum tend to represent significant business processes or some other kind of work that needs to be done. Not only do they encapsulate behaviors and rules, but also the information that is associated with those processes. Examples include: Pricing, Credit Check, Billing, and Fraud Analysis. A Pricing Business Component would encapsulate everything an

h e b b g e e e f e ef b e

organization needs to know about how to calculate the price of a product, including the product's base price (although this might belong in a Product component), discounts and rules for when they apply, and the calculation itself.

Detailed Description Text (1185):
FIG. 37 shows how a Billing Business Component 3700 may create an invoice. The control logic 3702 (i.e., the sequence of steps and business rules) associated with the billing process is encapsulated within the Billing component itself. The Billing component requests services from several entity-centric Business Components, but it also triggers Fraud Analysis 3704, a process-centric Business Component, if a specific business rule is satisfied. Note also that "Step 6" is performed within the Billing component itself. Perhaps this is where the invoice is created, reflecting the design team's decision to encapsulate the invoice within the Billing component. This is one valid approach. Another is to model a separate entity-centric Invoice component that encapsulates the concept of invoice. This would effectively decouple the invoice from the billing process which might be a good thing depending on the requiremients.

Detailed Description Text (1192):
A pattern is "an idea that has been useful in one practical context and will probably be useful in others." Think of them as blueprints, or designs for proven solutions to known problems. Having found the right pattern for a given problem, a developer must then apply it. Examples of patterns include: an analysis pattern for hierarchical relationships between organizations and/or people, a design pattern for maintaining an audit trail, a design pattern for applying different levels of security to different user types, and a design pattern for composite relationships between objects.

Detailed Description Text (1239):
However, they are looking into preferred customer cards. Furthermore, while analyzing the industry, the project team reads about a competitor with a pharmacy and video rental service. In both cases, customer information becomes critical. So the project team creates scenarios describing how they would use customer information to support these requirements. They create one Business Component Model that supports both today's and tomorrow's view of the customer.

Detailed Description Text (1278):
The close tie that component and object modeling enables between the software solution and business process may help software analysts and users or business analysts to better understand each other, reducing errors in communications. This represents a significant opportunity, because misunderstanding user requirements has been proven to be the most costly type of mistake in systems development. A component model further improves the understanding of the software design by providing a larger-grained model that is easier to digest.

Detailed Description Text (1330):
Systems development traditionally relies on a waterfall model. This approach manages development in sequential phases of activity such as analysis, design, code, and test. The waterfall provides control and discipline to development, particularly critical for large, mission-critical efforts.

Detailed Description Text (1341):
Most component-based methodologies focus primarily on analysis and design techniques. For example, less guidance is available for configuration management or testing. Yet, both of these aspects are more complex with component-based development, because of the greater level of granularity of the software decomposition. Because the methodologies are generic, they also typically do not address detailed architecture or design steps.

Detailed Description Text (1356):

h e b b g e e e f e ef b e

Components and objects are frequently considered to be equivalent technologies; however, they are not one in the same. While object-oriented systems may be developed using object-oriented analysis, design, and programming, a component-based system can be developed using a wide variety of languages, including procedural ones. As a result, the required depth of skills for a component-based project may depend on the blend of technologies used. For example, one project may require skills in COBOL, C++, and Smalltalk, while another may use Visual Basic exclusively. Because many projects are building components with objects, deep object-oriented skills may continue to be an essential ingredient in the success of a project.

Detailed Description Text (1391):
It is important to remember that many roles on the team are more demanding functionally than technically. Interviewing users, analyzing business processes, and designing the user interface all do not require extensive technical training. Moreover, not adequately understanding and analyzing the functional requirements are the most expensive mistakes. Research has shown that 70-80% of a system's mistakes result from misunderstood requirements.

Detailed Description Text (1393):
A component approach affects almost all aspects of the development lifecycle. For this reason the component learning curve cannot be equated with a programming learning curve such as `C`. There are multiple, distinct learning curves that affect individuals at many different levels in the organization: Component and object-oriented concepts and terminology Object analysis and design Programming language Programming environment and other development tools (e.g., browsers, debuggers, user interface tools) New architectures--such as how to use the project-specific application framework Management--such as estimating and planning for work, and managing iteration and prototyping

Detailed Description Text (1415):
A component-based development project requires creativity. The overall atmosphere is usually very challenging with fewer, concrete rules. The answer to many analysis and design decisions is, "it depends". Similarly, the development environments encourage exploration and browsing.

Detailed Description Text (1428):
Component-based development requires more time to scale the leaming curve, because it has multiple dimensions. Component technology skills cover a wide-range of competencies including analysis, design, programming, and management. Thus, leveraging expert mentors and skills, investing in adequate training, and ensuring continued support are all key to success.

Detailed Description Text (1471):
FIG. 47 illustrates a workcell organization approach including an activities component 4702, a credit/collections component 4704, a billing component 4706, and a finance component 4710. This approach combines the two previous approaches into a workcell. The primary orientation can be aligned either way, but a functional orientation seems more natural for a business application. A cell is comprised of a complete set of specialized skills such as functional analyst, object modeler, application architect, and even user. Cross-cell architects then provide specialized direction for a particular role.

Detailed Description Text (1491):
There often is not a direct mapping to the traditional roles that individuals expect. Analysts and Consultants may be given tasks with less creative freedom than they expect. For example, an Analyst role may involve less custom coding and more reusing, assembling, and testing of components. Design tasks for a new Consultant may also seem overly restrictive, because the challenge is to do things in a much more consistent, standard manner as dictated by the framework.

Detailed Description Text (1492):
On the other hand, because everything is often so new to the entire project team, in some ways everyone is starting together from scratch. Thus, in a few cases, very talented Analysts with prior component experience have assumed lead technical design roles.

Detailed Description Text (1503):
Systems development traditionally relies on a waterfall model. This approach manages development in sequential phases of activity such as analysis, design, code, and test. The waterfall model provides a controlled, orderly process for developing a system. Work is sequenced to ensure that the design addresses the correct requirements, implementation is based on upfront design, and system testing verifies and validates thoroughly unit tested components.

Detailed Description Text (1506):
Because of the above shortcomings, much of the OO and component community recommends some variation of iterative development, in which analysis, design, and coding activities overlap to some degree. A theme in these variations is the need to address risk by proceeding further in development sooner. Both the gained information and experience can influence the approach taken in the current phase.

Detailed Description Text (1515):
Distinguishing between a macro and a micro process provides a practical compromise. The pure, traditional waterfall has no distinction. There, the entire workplan and accompanying development approach sequence analyzing everything, then designing everything, then coding and testing everything, with no overlap. The same uniformity between macro and micro processes applies to a pure iterative model. In this case, the workplan reflects multiple iterations of the entire application. However, in either case, such extremism is not necessary. Instead, a plan can merge the two approaches by distinguishing between the: macro, high-level plan, and micro, phase or team-specific plan.

Detailed Description Text (1522):
Many decisions must then be considered from the vantage point of their ease of communication. This complicates iteration. For example, if analysis, design, and code overlap extensively, then by definition, requirements and design change later in the process. Communicating wide-scale changes late in development can be inefficient, wreaking havoc on existing code. Thus, iteration does not scale well to the macro level, because of communications overhead.

Detailed Description Text (1528):
Incremental development is often more palatable to managers than iterative development, because there is no explicit notion of repetition. Yet, the desirable benefits of iteration are often realized. For example, releasing consecutive versions of the system creates the opportunity, and often the requirement, to refine the initial release. The early implementation experience can also provide important productivity benefits for subsequent releases. This experience may also help drive out technical requirements for future releases, improving the analysis and design process.

Detailed Description Text (1532):
Even when incremental development does not prove feasible for entire application releases, the approach can be effective on a smaller scale. For example, the development and release of a single application may require extensive integration of diverse behaviors in a reusable domain component model. The domain components must be put in place early to allow reuse; then, behaviors are incrementally added as the business use cases are analyzed and designed. As in the previous case, iteration naturally occurs; but, again, incremental proves to be a more acceptable metaphor.

h e b b g e e e f e ef b e

Detailed Description Text (1538):
This model incorporates the idea of simultaneous top-down and bottom-up development. Much development effort may follow a relatively top-down, sequential approach. This includes analyzing and designing: the business environment and processes, domain model, and then application. Concurrently, an architecture effort proceeds bottom-up. This builds: the technology architecture of platform system software, hardware and infrastructure services; and then application architecture, or frameworks. Top-down and bottom-up efforts then conceptually meet in the middle, integrating the application framework with the application.

Detailed Description Text (1607):
Testing typically consumes anywhere from 50-80% of development effort. Despite this relative importance, testing receives little emphasis by component-based methodologies, which focus primarily on analysis and design techniques. This section presents testing lessons consistent with the primary themes in The Testing Process Practice Aid, produced by the Re-inventing Testing Project. These points merit increased emphasis, however, because experience has shown component-based systems increase testing complexity.

Detailed Description Text (1660):
On every component-based development project, teams spend time evaluating and establishing the environment in which analysts and developers create the deliverables. A workbench must be established that expedites the flow of deliverables through the different phases of the project. In component- and object-based solutions, these phases are very connected. This is largely because each subsequent phase tends to be an elaboration and refinement of the deliverables completed in previous phases. In addition, there is a strong desire to link deliverables and requirements from the earlier phases to the deliverables from the subsequent phases.

Detailed Description Text (1663):
To realize an environment that enhances the productivity of your analysts and programmers is a challenge for any project, but for projects building component-based solutions, it's even more difficult because of the technology's relative immaturity. You won't find any easy answers, yet.

Detailed Description Text (1669):
A development architecture should provide an environment for component-based solutions that supports a team through the Analysis, Design, and Construction phases of the development process. It should also serve as a productive environment for the on-going maintenance of an application. Conceptually it should integrate all of the necessary tools through an information model and most ideally through a central repository. The following are considerations that all component development architecture must consider. 1. Support Custom Process. The present invention uses a robust process for developing component-based solutions. It includes deliverables that are above and beyond the Unified Modeling Language (UML). Furthermore, projects often customize it. The environment must provide the ability to extend the information model (i.e., the meta-model). 2. Versioning & configuration management. The environment should provide the ability to version objects within the common information model at any level of granularity, keeping track of these changes over time. It should provide the same ability for composite objects (i.e., configurations of smaller objects). 3. Scalability. The repository-enabled environment must be able to support hundreds of users simultaneously, and hundreds of thousands of repository relationships. It should also scale downward, so that small project can use it. This is a major criterion for usability. 4. Query and impact analysis. As organizations begin to maintain their own component-based assets, they must be able to analyze the impact of change requests (e.g., where-used searches). The ability to trace requirements is also critical. 5. Asset catalog (reuse). As organizations begin to reuse existing assets, it may become

h e b b g e e e f e ef b e

increasingly important to provide a catalog of components, frameworks, patterns, etc. The catalog should make it possible to search for relevant assets in a wide variety of ways. It should also provide a means for applying frameworks and patterns. 6. Code generation. The ability to generate the application structure from the model is essential to high productivity. Furthermore, this step should be transparent to the user. As far as the user is concerned, a change to the model is a change to the code. 7. Desktop Tool Integration. The repository-enabled environment must provide integration between all desktop tools (e.g., MS Office, Visio, OO CASE tools, designers, etc.) through component object models such as ActiveX. In addition, these tools must have access to the common open information models. 8. Non-redundant storage. The environment should avoid redundant storage of information, whenever possible. Everything from training to documentation to active components should be automatically updated or notified of changes. 9. Multiple users and locations. Many users may need access to the environment during the course of a development effort. Furthermore, because one supports global communities of practice, there is a strong need to share this information securely and across disparate locations.

Detailed Description Text (1738):
Query & Impact Analysis

Detailed Description Text (1765):
An old saying goes, "Cheap, fast and good--I'll give you two out of three". Many of clients may react negatively to this philosophy, because they would certainly like excellence in all three areas. Yet, the fact remains that difficult tradeoffs exist between performance, quality, and the cost of the system. For example, no one intentionally designs a slow system. Thus, it is critical to define performance goals in business terms based on cost/benefit analysis.

Detailed Description Text (1814):
Opportunities for performance tuning are found both in bottlenecks and in distributed inefficiencies. There are generally many tools available in detecting bottlenecks. Distributed inefficiencies are usually more difficult to identify with tools. Whether performance optimizations are realized through cognitive analysis, or tool-assisted profiling, it is important to measure the gains against a baseline performance level.

Detailed Description Text (1906):
As is stated in the Component Technology Architecture Framework, "Business components are the core of any application, they represent concepts within the business domain. They encapsulate the information and behavior associated with those concepts. Examples of business components include: Customer, Product, Order, Inventory, Pricing, Credit Check, Billing, and Fraud Analysis." These are the components that in many cases have been the most elusive for reuse but hold the highest promise for attacking the cost of development. In this area there are at least three targeted categories of business components, Common Business Components, Common Business Services and Common Business Facilities.

Detailed Description Text (2505):
Benefits Requirements Traceability. Exceptions requirements are captured and managed through implementation. Hierarchy Design. Analysis may show optimizations that can be made such as handling a subtree of exceptions with the same code, as the response is the same to any exception in the subtree. Interface Design. Discovery of interface requirements on the exception classes to support a particular response is another benefit. Handler design. Assists in exception handling design by identifying common responses that can be leveraged by the handlers.

Detailed Description Text (2905):
Although only a few embodiments of the present invention have been described in

h e b b g e e e f e ef b e

detail herein, it should be understood that the present invention may be embodied in many other specific forms without departing from the spirit or scope of the invention. Therefore, the present examples and embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope of the appended claims.

Detailed Description Paragraph Table (3):
Category Basic Func Adv Analysis and Design 4 wks 6-8 mos. 18-24 mos Implementation 3-4 wks 5-6 mos 18-24 mos Frameworks Design 16 wks 12-24 mos 24-48 mos Management 3-4 wks 12-18 mos 24-36 mos

Current US Original Classification (1):
709/228

Other Reference Publication (2):
Kovalerchuck et al., comparison of relational methods and attribute-based methods for data mining in intelligent systems, proceedings of the 1999 IEEE, International Symposium on Intelligent Systems and Semiotics, Cambridge, MA, PP 162-166. Date Sep. 1999.



---

☐ 4. Document ID: US 6640244 B1

L24: Entry 4 of 10 File: USPT Oct 28, 2003

DOCUMENT-IDENTIFIER: US 6640244 B1
TITLE: Request batcher in a transaction services patterns environment

Detailed Description Text (4):
OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.

Detailed Description Text (34):
Architecture also is an engineering discipline. It creates and also depends on a structured manner to analyze and design whatever is to be built. Like all living disciplines, architecture continues to grow and evolve. Engineering discoveries move the field forward. Certain design and engineering principles clearly show themselves to be successful in practice, and these then become repeatable components of additional work. The ability to continue to master each component, as well as the interrelations among components, is a distinguishing characteristic of architecture.

Detailed Description Text (39):
Finally, architecture must be understood as a process 200, not just a thing. This process can be described at a very high level using FIG. 2. Step 1: Analyze 202. The architect must begin by listening to and researching the needs of the client.

h e b b g e e e f e ef b e

What is the function of the building?What is its environment? What are the limitations set by budget and use? Step 2: Design 204. This is a blueprint stage. The architect creates one or several designs showing the layout of the structure, how different spaces fit together, how everything looks from different views, what materials are to be used, and so forth. Step 3: Model & Test 206. Not every architectural project has this step, but in many cases, the architect will create a scale model/prototype of the finished product, allowing the client a clearer sense of what the ultimate solution will look like. A model is a kind of test stage, allowing everyone to test the design in a near-real-life setting. Step 4: Build 208. This is the actual construction of the building, in general accord with the blueprints and prototype. Step 5: Operate and Evolve 210. The building is to be lived in and used, of course, and so an important step is to ensure that the finished product is tended and operated effectively. Architects themselves may not be involved in the operation of their building, but they certainly would be involved in future expansions or evolutions of the building. Stewart Brand's recent text, How Buildings Learn, argues that effective architecture takes into account the fact that buildings "learn": as people live and work in them over time, those people will seek to alter the building in subtle, or not so subtle, ways.

Detailed Description Text (81):
The use of architecture frameworks during analysis and design can reduce the risks of an IT solution. It should improve development productivity through reuse, as well as the IT solution's reliability and maintainability.

Detailed Description Text (143):
Existing Architecture and Infrastructure 700 E1. Other Netcentric applications been developed and placed in production. The user community is often less resistant to accept the use of new technology to address changing business drivers if they are not completely unfamiliar with the characteristics of the technology. If an application based on a Netcentric architecture has already been successfully piloted or deployed, acceptance of additional systems will be eased. E2. The client has significant technology skills within its IT department. This is especially important if the client plans on developing or operating the application themselves. A significant investment in training and changes to internal organizations may be necessary for successful deployment of this type of system. The client must have a culture that supports change. Some organizations are very conservative and strong, making it difficult to deliver a successful project using new technology. E3. The client has multiple hardware/operating system configurations for their client machines. In traditional client/server environments, distributing an application internally or externally for an enterprise requires that the application be ported, recompiled and tested for all specific workstation operating systems. Use of a Universal Client or web-browser may eliminate many of these problems by providing a consistent and familiar user interface on many different operating systems and hardware platforms. E4. The application will run on a device other than a PC. The momentum of the Internet is putting a lot of pressure on vendors of various devices to be web-enabled. Having the Internet infrastructure in place makes it more feasible for vendors to create new physical devices from which electronic information can be accessed. For example, Web televisions are gaining momentum. Now users can access the Internet from a television set. Network Computers, thin-client devices that download and run applications from a centrally maintained server are generating a lot of interest. Also, users want to have access to the same information from multiple physical devices. For example, a user might want to have access to his/her e-mail from a cellular phone, from a Web TV or their portable PC. E5. The current legacy systems can scale to serve a potentially large new audience. Expanding the user community of a legacy host or client/server system by including an audience which is external to the company can result in dramatic increases in system usage. The additional demand and increased usage placed on existing legacy systems is often difficult to estimate or predict. Analysis must be conducted to ensure existing legacy systems and infrastructure can absorb this increase.

h e b b g e e e f e ef b e

Detailed Description Text (533):
In summary, the Directory service performs the following functions: Stores information about network resources and users and tracks relationships Organizes resource access information in order to aid resources in locating and accessing other resources throughout the network Provides location transparency, since resources are accessed through a directory rather than based on their physical location Converts between logical resource names and physical resource addresses Interacts with Security services such as authentication and authorization track identities and permissions Provides single network logon to file and print resources; can provide single network logon for network applications that are integrated with the Directory service Distributes directory information throughout the enterprise (for reliability and location-independent access) Synchronizes multiple directory databases Enables access to heterogeneous systems (integration of various network operating systems, platforms, etc.)

Detailed Description Text (605):
The following are examples of File Transfer products: Computer Associates' CA-XCOM--provides data transport between mainframes, midrange, UNIX, and PC systems. XcelleNet's RemoteWare--retrieves, appends, copies, sends, deletes, and renames files between remote users and enterprise systems. Hewlett-Packard's HP FTAM--provides file transfer, access, and management of files in OSI networks.

Detailed Description Text (751):
Authentication can occur through various means: Basic Authentication--requires that the Web client supply a user name and password before servicing a request. Basic Authentication does not encrypt the password in any way, and thus the password travels in the clear over the network where it could be detected with a network sniffer program or device. Basic authentication is not secure enough for banking applications or anywhere where there may be a financial incentive for someone to steal someone's account information. Basic authentication is however the easiest mechanism to setup and administer and requires no special software at the Web client. ID/Password Encryption--offers a somewhat higher level of security by requiring that the user name and password be encrypted during transit. The user name and password are transmitted as a scrambled message as part of each request because there is no persistent connection open between the Web client and the Web server. Digital Certificates or Signatures--encrypted digital keys that are issued by a third party "trusted" organization (i.e. Verisign); used to verify user's authenticity. Hardware tokens--small physical devices that may generate a one-time password or that may be inserted into a card reader for authentication purposes. Virtual tokens--typically a file on a floppy or hard drive used for authentication (e.g. Lotus Notes ID file). Biometric identification--the analysis of biological characteristics to verify individuals identify (e.g., fingerprints, voice recognition, retinal scans).

Detailed Description Text (813):
QoS can be achieved in various ways as listed below: Specialized QoS Communications Protocols--provide guaranteed QoS. Asynchronous Transfer Mode (ATM)--ATM is a connection-oriented wide area and local area networking protocol that delivers QoS on a per-connection basis. QoS is negotiated as part of the initial connection set up and as network conditions change. Because of the small size of ATM data cells, QoS can be better managed, compared to protocols such as Ethernet that have large frames that can tie up network components. For ATM to deliver QOS to applications, ATM must be used end-to-end. Resource Reservation Protocol (RSVP)--The emerging RSVP specification, proposed by the Internet Engineering Task Force (IETF), allows applications to reserve router bandwidth for delay-sensitive IP traffic. With RSVP, QoS is negotiated for each application connection. RSVP enables the network to reserve resources from end to end, using Frame Relay techniques on Frame Relay networks, ATM techniques on ATM, and so on. In this way, RSVP can achieve QoS across a variety of network technologies, as long as all intermediate nodes are

RSVP-capable. IP Stream Switching--improves network performance but does not guarantee QoS. IP Switching--IP Switching is an emerging technology that can increase network throughput for streams of data by combining IP routing software with ATM switching hardware. With IP Switching, an IP switch analyzes each stream of packets directed from a single source to a specific destination, and classifies it as short- or long-lived. Long-lived flows are assigned ATM Virtual Channels (VCS) that bypass the IP router and move through the switching fabric at the full ATM line speed. Short-lived flows continue to be routed through traditional store-and-forward transfer. Tag Switching--Like IP Switching, emerging Tag Switching technology also improves network throughput for IP data streams. Tag Switching aggregates one or more data streams destined for the same location and assigns a single tag to all associated packets. This allows routers to more efficiently transfer the tagged data. Tag Switching is also known as Multiprotocol Label Switching. Data Prioritization--improves network performance but does not guarantee QoS. While not an example of end-to-end QoS, various network components can be configured to prioritize their handling of specified types of traffic. For example, routers can be configured to handle legacy mainframe traffic (SNA) in front of other traffic (e.g., TCP/IP). A similar technique is the use of prioritized circuits within Frame Relay, in which the Frame Relay network vendor assigns different priorities to different permanent virtual circuits. Prioritization techniques are of limited effectiveness if data must also pass through network components that are not configured for prioritization (e.g., network components run by third party network providers).

Detailed Description Text (963):
Logging must be done, however to mitigate problems, centralize logs and create a standard, usable log format. 3rd party logs should be mapped into the central format before any analysis is attempted.

Detailed Description Text (1088):
Workflow can be further divided into the following components: Role management Role management ie provides for the assignment of tasks to roles which can then be mapped to individuals. A role defines responsibilities which are required in completing a business process. A business worker must be able to route documents and folders to a role, independent of the specific person, or process filling that role. For example, a request is routed to a supervisor role or to Purchasing, rather than to "Mary" or "Tom." If objects are routed to Mary and Mary leaves the company or is reassigned, a new recipient under a new condition would have to be added to an old event. Roles are also important when a number of different people have the authority to do the same work, such as claims adjusters; just assign the request to the next available person. In addition, a process or agent can assume a role; it doesn't need to be a person. Role Management Services provide this additional level of directory indirection. Route management Route management enables the routing of tasks to the next role, which can be done in the following ways: Serial--the tasks are sequentially performed; Parallel--the work is divided among different players; Conditional--routing is based upon certain conditions; and Ad hoc--work which is not part of a predefined process. Workflow routing services route "work" to the appropriate workflow queues. When an application completes processing a task, it uses these services to route the work-in-progress to the next required task or tasks and, in some cases, notify interested parties of the resulting work queue changes. The automatic movement of information and control from one workflow step to another requires work profiles that describe the task relationships for completing various business processes. The concept of Integrated Performance Support can be exhibited by providing user access to these work profiles. Such access can be solely informational--to allow the user to understand the relationship between tasks, or identify which tasks need to be completed for a particular work flow--or navigational--to allow the user to move between tasks. Route Management Services also support the routing and delivery of necessary information (e.g., documents, data, forms, applications, etc.) to the next step in the work flow as needed. Rule Management A business process workflow is typically

composed of many different roles and routes. Decisions must be made as to what to route to which role, and when. Rule Management Services support the routing of workflow activities by providing the intelligence necessary to determine which routes are appropriate given the state of a given process and knowledge of the organization's workflow processing rules. Rule Management Services are typically implemented through easily maintainable tables or rule bases which define the possible flows for a business event. Queue Management These services provide access to the workflow queues which are used to schedule work. In order to perform workload analysis or to create "to do lists" for users, an application may query these queues based on various criteria (a business event, status, assigned user, etc.). in addition, manipulation services are provided to allow queue entries to be modified. Workflow services allow users and management to monitor and access workflow queue information and to invoke applications directly.

Detailed Description Text (1112):
Issues to consider include the following: (1) samples and assists that are available to the developer; (2) existence of a scripting or programming language; (3) granularity of the security, or in other words, at what levels can security be added; (4) freedom of choosing productivity applications; (5) existence of aggregate functions which allow for analysis of the workflow efficiency; (6) existence/need for Business Processing Re-engineering tools.

Detailed Description Text (1156):
Objects are an easy metaphor to understand and manage. There are still substantial risks involved, particularly because component- and object-orientation has a pervasive impact on areas as broad as analysis and design, planning, and development tools.

Detailed Description Text (1159):
Component and object technology impacts most aspects of software development and management. Component technology is a new technology and a driving influence in the evolution of object-oriented (OO) methodologies. The Management Considerations section of the Introduction to Component-Based Development uses the Business Integration (BI) Model to discuss the impact of OO, including: Strategy and planning with a long-term view towards building reusable, enterprise software assets. Technology and architecture approaches for building cohesive, loosely coupled systems that provide long-term flexibility. Processes that shift analysis/design techniques from functional, procedural decomposition to business process modeling. These techniques are then used to decompose the system into domain objects and processes. People and organization strategies that emphasize greater specialization of skills within structures that support inter-team collaboration.

Detailed Description Text (1162):
Many of these considerations have been addressed over the last few years. Most published literature continues to focus on narrow technology issues, such as programming techniques or generic methodologies, such as analysis and design approaches or notation. Still, a growing number of publications and vendor strategies attack the enterprise needs within on-line netcentric execution models. Real-world, client solutions involve making pragmatic decisions, in which compromise occurs at the intersection of the four major OO themes. Experience with many component client projects in diverse industries uniquely positions a user to effectively address these complexities.

Detailed Description Text (1180):
Business Components represent real-world concepts in the business domain. They encapsulate everything about those concepts including name, purpose, knowledge, behavior, and all other intelligence. Examples include: Customer, Product, Order, Inventory, Pricing, Credit Check, Billing, and Fraud Analysis. One might think of a Business Component as a depiction or portrait of a particular business concept, and

h e b b g e e e f e ef b e

as a whole, the Business Component Model is a depiction or portrait of the entire business. It's also important to note that although this begins the process of defining the application architecture for a set of desired business capabilities, the applicability of the Business Component Model extends beyond application building.

Detailed Description Text (1185):
In the Business Architecture stage 3604, a project team begins to define the application architecture for an organization's business capabilities using Business Components. Business Components model real-world concepts in the business domain (e.g., customers, products, orders, inventory, pricing, credit check, billing, and fraud analysis). This is not the same as data modeling because Business Components encapsulate both information and behavior. At this point in the process, an inventory of Business Components is sufficient, along with a definition, list of entities, and list of responsibilities for each Business Component.

Detailed Description Text (1186):
In Capability Analysis 3606 and the first part of Capability Release Design 3608, the project team designs Business Components in more detail, making sure they satisfy the application requirements. The team builds upon its previous work by providing a formal definition for each Business Component, including the services being offered. Another name for these services is "Business Component Interfaces." The team also models the interactions between Business Components.

Detailed Description Text (1194):
Business Components on the process-centric side of the spectrum tend to represent significant business processes or some other kind of work that needs to be done. Not only do they encapsulate behaviors and rules, but also the information that is associated with those processes. Examples include: Pricing, Credit Check, Billing, and Fraud Analysis. A Pricing Business Component would encapsulate everything an organization needs to know about how to calculate the price of a product, including the product's base price (although this might belong in a Product component), discounts and rules for when they apply, and the calculation itself.

Detailed Description Text (1198):
FIG. 37 shows how a Billing Business Component 3700 may create an invoice. The control logic 3702 (i.e., the sequence of steps and business rules) associated with the billing process is encapsulated within the Billing component itself. The Billing component requests services from several entity-centric Business Components, but it also triggers Fraud Analysis 3704, a process-centric Business Component, if a specific business rule is satisfied. Note also that "Step 6" is performed within the Billing component itself. Perhaps this is where the invoice is created, reflecting the design team's decision to encapsulate the invoice within the Billing component. This is one valid approach. Another is to model a separate entity-centric Invoice component that encapsulates the concept of invoice. This would effectively decouple the invoice from the billing process which might be a good thing depending on the requirements.

Detailed Description Text (1205):
A pattern is "an idea that has been useful in one practical context and will probably be useful in others." Think of them as blueprints, or designs for proven solutions to known problems. Having found the right pattern for a given problem, a developer must then apply it. Examples of patterns include: an analysis pattern for hierarchical relationships between organizations and/or people, a design pattern for maintaining an audit trail, a design pattern for applying different levels of security to different user types, and a design pattern for composite relationships between objects.

Detailed Description Text (1253):
However, they are looking into preferred customer cards. Furthermore, while

analyzing the industry, the project team reads about a competitor with a pharmacy and video rental service. In both cases, customer information becomes critical. So the project team creates scenarios describing how they would use customer information to support these requirements. They create one Business Component Model that supports both today's and tomorrow's view of the customer.

Detailed Description Text (1292):
The close tie that component and object modeling enables between the software solution and business process may help software analysts and users or business analysts to better understand each other, reducing errors in communications. This represents a significant opportunity, because misunderstanding user requirements has been proven to be the most costly type of mistake in systems development. A component model further improves the understanding of the software design by providing a larger-grained model that is easier to digest.

Detailed Description Text (1344):
Systems development traditionally relies on a waterfall model. This approach manages development in sequential phases of activity such as analysis, design, code, and test. The waterfall provides control and discipline to development, particularly critical for large, mission-critical efforts.

Detailed Description Text (1355):
Most component-based methodologies focus primarily on analysis and design techniques. For example, less guidance is available for configuration management or testing. Yet, both of these aspects are more complex with component-based development, because of the greater level of granularity of the software decomposition. Because the methodologies are generic, they also typically do not address detailed architecture or design steps.

Detailed Description Text (1370):
Components and objects are frequently considered to be equivalent technologies; however, they are not one in the same. While object-oriented systems may be developed using object-oriented analysis, design, and programming, a component-based system can be developed using a wide variety of languages, including procedural ones. As a result, the required depth of skills for a component-based project may depend on the blend of technologies used. For example, one project may require skills in COBOL, C++, and Smalltalk, while another may use Visual Basic exclusively. Because many projects are building components with objects, deep object-oriented skills may continue to be an essential ingredient in the success of a project.

Detailed Description Text (1402):
It is important to remember that many roles on the team are more demanding functionally than technically. Interviewing users, analyzing business processes, and designing the user interface all do not require extensive technical training. Moreover, not adequately understanding and analyzing the functional requirements are the most expensive mistakes. Research has shown that 70-80% of a system's mistakes result from misunderstood requirements.

Detailed Description Text (1404):
A component approach affects almost all aspects of the development lifecycle. For this reason the component learning curve cannot be equated with a programming learning curve such as `C`. There are multiple, distinct learning curves that affect individuals at many different levels in the organization: Component and object-oriented concepts and terminology Object analysis and design Programming language Programming environment and other development tools (e.g., browsers, debuggers, user interface tools) New architectures--such as how to use the project-specific application framework Management--such as estimating and planning for work, and managing iteration and prototyping

Detailed Description Text (1426):
A component-based development project requires creativity. The overall atmosphere is usually very challenging with fewer, concrete rules. The answer to many analysis and design decisions is, "it depends". Similarly, the development environments encourage exploration and browsing.

Detailed Description Text (1439):
Component-based development requires more time to scale the learning curve, because it has multiple dimensions. Component technology skills cover a wide-range of competencies including analysis, design, programming, and management. Thus, leveraging expert mentors and skills, investing in adequate training, and ensuring continued support are all key to success.

Detailed Description Text (1482):
FIG. 47 illustrates a workcell organization approach including an activities component 4702, a credit/collections component 4704, a billing component 4706, and a finance component 4710. This approach combines the two previous approaches into a workcell. The primary orientation can be aligned either way, but a functional orientation seems more natural for a business application. A cell is comprised of a complete set of specialized skills such as functional analyst, object modeler, application architect, and even user. Cross-cell architects then provide specialized direction for a particular role.

Detailed Description Text (1501):
There often is not a direct mapping to the traditional roles that individuals expect. Analysts and Consultants may be given tasks with less creative freedom than they expect. For example, an Analyst role may involve less custom coding and more reusing, assembling, and testing of components. Design tasks for a new Consultant may also seem overly restrictive, because the challenge is to do things in a much more consistent, standard manner as dictated by the framework.

Detailed Description Text (1502):
On the other hand, because everything is often so new to the entire project team, in some ways everyone is starting together from scratch. Thus, in a few cases, very talented Analysts with prior component experience have assumed lead technical design roles.

Detailed Description Text (1512):
Systems development traditionally relies on a waterfall model. This approach manages development in sequential phases of activity such as analysis, design, code, and test. The waterfall model provides a controlled, orderly process for developing a system. Work is sequenced to ensure that the design addresses the correct requirements, implementation is based on upfront design, and system testing verifies and validates thoroughly unit tested components.

Detailed Description Text (1515):
Because of the above shortcomings, much of the OO and component community recommends some variation of iterative development, in which analysis, design, and coding activities overlap to some degree. A theme in these variations is the need to address risk by proceeding further in development sooner. Both the gained information and experience can influence the approach taken in the current phase.

Detailed Description Text (1524):
Distinguishing between a macro and a micro process provides a practical compromise. The pure, traditional waterfall has no distinction. There, the entire workplan and accompanying development approach sequence analyzing everything, then designing everything, then coding and testing everything, with no overlap. The same uniformity between macro and micro processes applies to a pure iterative model. In this case, the workplan reflects multiple iterations of the entire application. However, in either case, such extremism is not necessary. Instead, a plan can merge

h e b b g e e e f e ef b e

the two approaches by distinguishing between the: macro, high-level plan, and micro, phase or team-specific plan.

Detailed Description Text (1531):
Many decisions must then be considered from the vantage point of their ease of communication. This complicates iteration. For example, if analysis, design, and code overlap extensively, then by definition, requirements and design change later in the process. Communicating wide-scale changes late in development can be inefficient, wreaking havoc on existing code. Thus, iteration does not scale well to the macro level, because of communications overhead.

Detailed Description Text (1537):
Incremental development is often more palatable to managers than iterative development, because there is no explicit notion of repetition. Yet, the desirable benefits of iteration are often realized. For example, releasing consecutive versions of the system creates the opportunity, and often the requirement, to refine the initial release. The early implementation experience can also provide important productivity benefits for subsequent releases. This experience may also help drive out technical requirements for future releases, improving the analysis and design process.

Detailed Description Text (1541):
Even when incremental development does not prove feasible for entire application releases, the approach can be effective on a smaller scale. For example, the development and release of a single application may require extensive integration of diverse behaviors in a reusable domain component model. The domain components must be put in place early to allow reuse; then, behaviors are incrementally added as the business use cases are analyzed and designed. As in the previous case, iteration naturally occurs; but, again, incremental proves to be a more acceptable metaphor.

Detailed Description Text (1547):
This model incorporates the idea of simultaneous top-down and bottom-up development. Much development effort may follow a relatively top-down, sequential approach. This includes analyzing and designing: the business environment and processes, domain model, and then application. Concurrently, an architecture effort proceeds bottom-up. This builds: the technology architecture of platform system software, hardware and infrastructure services; and then application architecture, or frameworks. Top-down and bottom-up efforts then conceptually meet in the middle, integrating the application framework with the application.

Detailed Description Text (1616):
Testing typically consumes anywhere from 50-80% of development effort. Despite this relative importance, testing receives little emphasis by component-based methodologies, which focus primarily on analysis and design techniques. This section presents testing lessons consistent with the primary themes in The Testing Process Practice Aid, produced by the Re-inventing Testing Project. These points merit increased emphasis, however, because experience has shown component-based systems increase testing complexity.

Detailed Description Text (1669):
On every component-based development project, teams spend time evaluating and establishing the environment in which analysts and developers create the deliverables. A workbench must be established that expedites the flow of deliverables through the different phases of the project. In component- and object-based solutions, these phases are very connected. This is largely because each subsequent phase tends to be an elaboration and refinement of the deliverables completed in previous phases. In addition, there is a strong desire to link deliverables and requirements from the earlier phases to the deliverables from the subsequent phases.

h e b b g e e e f e ef b e

Detailed Description Text (1673):
To realize an environment that enhances the productivity of your analysts and programmers is a challenge for any project, but for projects building component-based solutions, it's even more difficult because of the technology's relative immaturity. You won't find any easy answers, yet.

Detailed Description Text (1679):
A development architecture should provide an environment for component-based solutions that supports a team through the Analysis, Design, and Construction phases of the development process. It should also serve as a productive environment for the on-going maintenance of an application. Conceptually it should integrate all of the necessary tools through an information model and most ideally through a central repository. The following are considerations that all component development architecture must consider. 1. Support Custom Process. The present invention uses a robust process for developing component-based solutions. It includes deliverables that are above and beyond the Unified Modeling Language (UML). Furthermore, projects often customize it. The environment must provide the ability to extend the information model (i.e., the meta-model). 2. Versioning & configuration management. The environment should provide the ability to version objects within the common information model at any level of granularity, keeping track of these changes over time. It should provide the same ability for composite objects (i.e., configurations of smaller objects). 3. Scalability. The repository-enabled environment must be able to support hundreds of users simultaneously, and hundreds of thousands of repository relationships. It should also scale downward, so that small project can use it. This is a major criterion for usability. 4. Query and impact analysis. As organizations begin to maintain their own component-based assets, they must be able to analyze the impact of change requests (e.g., where-used searches). The ability to trace requirements is also critical. 5. Asset catalog (reuse). As organizations begin to reuse existing assets, it may become increasingly important to provide a catalog of components, frameworks, patterns, etc. The catalog should make it possible to search for relevant assets in a wide variety of ways. It should also provide a means for applying frameworks and patterns. 6. Code generation. The ability to generate the application structure from the model is essential to high productivity. Furthermore, this step should be transparent to the user. As far as the user is concerned, a change to the model is a change to the code. 7. Desktop Tool Integration. The repository-enabled environment must provide integration between all desktop tools (e.g., MS Office, Visio, OO CASE tools, designers, etc.) through component object models such as ActiveX. In addition, these tools must have access to the common open information models. 8. Non-redundant storage. The environment should avoid redundant storage of information, whenever possible. Everything from training to documentation to active components should be automatically updated or notified of changes. 9. Multiple users and locations. Many users may need access to the environment during the course of a development effort. Furthermore, because one supports global communities of practice, there is a strong need to share this information securely and across disparate locations.

Detailed Description Text (1745):
Query & Impact Analysis

Detailed Description Text (1772):
An old saying goes, "Cheap, fast and good--I'll give you two out of three". Many of clients may react negatively to this philosophy, because they would certainly like excellence in all three areas. Yet, the fact remains that difficult tradeoffs exist between performance, quality, and the cost of the system. For example, no one intentionally designs a slow system. Thus, it is critical to define performance goals in business terms based on cost/benefit analysis.

Detailed Description Text (1821):

h e b b g e e e f e ef b e

Opportunities for performance tuning are found both in bottlenecks and in distributed inefficiencies. There are generally many tools available in detecting bottlenecks. Distributed inefficiencies are usually more difficult to identify with tools. Whether performance optimizations are realized through cognitive analysis, or tool-assisted profiling, it is important to measure the gains against a baseline performance level.

Detailed Description Text (1912):
As is stated in the Component Technology Architecture Framework, "Business components are the core of any application, they represent concepts within the business domain. They encapsulate the information and behavior associated with those concepts. Examples of business components include: Customer, Product, Order, Inventory, Pricing, Credit Check, Billing, and Fraud Analysis." These are the components that in many cases have been the most elusive for reuse but hold the highest promise for attacking the cost of development. In this area there are at least three targeted categories of business components, Common Business Components, Common Business Services and Common Business Facilities.

Detailed Description Text (2514):
Benefits Requirements Traceability. Exceptions requirements are captured and managed through implementation. Hierarchy Design. Analysis may show optimizations that can be made such as handling a subtree of exceptions with the same code, as the response is the same to any exception in the subtree. Interface Design. Discovery of interface requirements on the exception classes to support a particular response is another benefit. Handler design. Assists in exception handling design by identifying common responses that can be leveraged by the handlers.

Detailed Description Text (2912):
Although only a few embodiments of the present invention have been described in detail herein, it should be understood that the present invention may be embodied in many other specific forms without departing from the spirit or scope of the invention. Therefore, the present examples and embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope of the appended claims.

Detailed Description Paragraph Table (3):
Category Basic Func Adv Analysis and Design 4 wks 6-8 mos. 18-24 mos Implementation 3-4 wks 5-6 mos 18-24 mos Frameworks Design 16 wks 12-24 mos 24-48 mos Management 3-4 wks 12-18 mos 24-36 mos

Current US Original Classification (1):
709/207

Current US Cross Reference Classification (1):
707/10

Other Reference Publication (2):
Kovalerchuck et al., 'comparison of relational methods and attribute-based methods for data mining in intelligent systems, proceedings of the 1999 IEEE, International Symposium on Intelligent Systems and Semiotics, Cambridge, MA, PP 162-166. Date Sep. 1999.



---

☐ 5. Document ID: US 6640238 B1

h e b b g e e e f e ef b e